F. Les opérateurs & fonctions de base

1. Opérateurs arithmétiques

Définition

- Comme on l'a déjà vu, l'affectation est effectuée par l'opérateur **:=**
- Les opérateurs arithmétiques **+ - *** et **/** ont les mêmes fonctions qu'ils ont dans les autres langages de programmation, sauf pour l'opérateur ****** qui signifie l'exposant : **A ** B** ⟺ A^B

Exemple

```
DECLARE
      x NUMBER := 4;
      y NUMBER;
BEGIN
      y := (x + 1) * 2;
      dbms_output.put_line('produit:      ' || y);
      y := (x + 1) ** 2;
      dbms_output.put_line('puissance: ' || y);
END;
```

Message | DBMS Output

```
produit:    10
puissance: 25
```

2. Fonctions arithmétiques

Définition	
`ABS(n)`	La valeur absolue de n
`CEIL(n)`	L'entier le plus petit qui soit supérieur ou égal à n
`FLOOR(n)`	L'entier le plus grand qui soit inférieur ou égal à n
`GREATEST(n1,n2,n3)`	La valeur la plus grande entre n1, n2 et n3
`LEAST(n1,n2,n3)`	La valeur la plus petite entre n1, n2 et n3
`REMAINDER(n,m)`	Le reste de la division euclidienne de n par m (*modulo*)
`POWER(n,m)`	n puissance m
`SQRT(n)`	La racine carrée de n (*square root*)

Exemple

```
DECLARE
      x NUMBER;
      y NUMBER;
      z NUMBER;
      r NUMBER(3);
      r1 NUMBER(2,1);
BEGIN
      x := 1;
      y := 2;
      z := 3;
      r := "SQRT"((x + y) ** y);
      dbms_output.put_line('resultat 1:  ' || r);
      r := "POWER"(r, "POWER"(y, y));
      dbms_output.put_line('resultat 2:  ' || r);
      r := "REMAINDER"(r + 1, "GREATEST"(x, y, z));
```

S.ELYAHYAOUI PLSQL

```
      dbms_output.put_line('resultat 3:  ' || r);
      r1 := z / y;
      dbms_output.put_line('resultat 4:  ' || "FLOOR"(r1));
      dbms_output.put_line('resultat 5:  ' || "CEIL"(r1));
END;
```

```
Message | DBMS Output
resultat 1: 3
resultat 2: 81
resultat 3: 1
resultat 4: 1
resultat 5: 2
```

3. Manipuler des dates

a. Conversion date → chaine avec formatage

Définition : Fonction *TO_CHAR*

Cette fonction prédéfinie en PL/SQL permet de convertir une date en une chaine de caractères, avec optionnellement un format de sortie, spécifié en 2ieme paramètre. (*Les fonctions seront revues en détail dans les prochains paragraphes*)

Syntaxe :

```
"TO_CHAR"(uneDate, 'unFormat');
```

Formats de sortie :

Année en **4** chiffres :	**YYYY**
Année en **2** chiffres :	**YY**
Mois en **chiffres** :	**MM**
Mois en **lettres** :	**MONTH**
Jour en **chiffres** :	**DD**
Jour en **lettres** :	**DAY**
Heures en format **12h** :	**HH**
Heures en format **24h** :	**HH24**
Minutes :	**MI**

Exemple

```
DECLARE
      d DATE;
      v VARCHAR2(50);
BEGIN
      d := SYSDATE;
      dbms_output.put_line(d);
      v := "TO_CHAR"(d, 'DAY DD MONTH YYYY');
      dbms_output.put_line(v);
      v := "TO_CHAR"(d, 'DD-MONTH');
      dbms_output.put_line(v);
      v := "TO_CHAR"(d, 'DD/MM/YY  HH24:MI');
      dbms_output.put_line(v);
END;
```

S.ELYAHYAOUI - PL/SQL

Message | DBMS Output

```
03-MAR-15
TUESDAY  03 MARCH    2015
03-MARCH
03/03/15  16:15
```



IMPORTANT

Le format par défaut d'une date est **DD-MON-YY**.
Une déclaration de variable de type **DATE** qui ne respecte pas cette spécification conduit à une erreur d'exécution.

Exemple

```
DECLARE
      d DATE;
BEGIN
      d := '22-01-2010';
      dbms_output.put_line(d);
END;
```

Message

```
[SQL] DECLARE
      d DATE;
BEGIN
      d := '22-01-2010';
      dbms_output.put_line(d);
END;
[Err] ORA-01843: not a valid month
ORA-06512: at line 4
```

b. Fonctions sur les dates

Définition	
ADD_MONTHS(date,n)	Retourne la date + n mois, n peut être positif ou négatif.
DBTIMEZONE()	Retourne le fuseau horaire actuel (Ex. : *+00:00, -01:00, …*)
SYSDATE() ou CURRENT_DATE()	Retourne la date actuelle.
NEXT_DAY(date, jour)	Retourne le prochain jour de la semaine, qui correspond au jour spécifié en 2ième paramètre, à partir de la date spécifiée en 1ier paramètre.
MONTHS_BETWEEN(date1, date2)	Retourne le nombre de mois entre date1 et date2. (date1 – date2)

Exemple

```
DECLARE
      d1 DATE;
      d2 DATE;
BEGIN
      d1 := '01-JAN-2001';
      d2 := "ADD_MONTHS"(d1, 12);
      dbms_output.put_line(d2);
      d2 := "ADD_MONTHS"(d1, -2);
      dbms_output.put_line("TO_CHAR"(d2, 'DD/MM/YYYY'));
```

```
      d1 := "CURRENT_DATE"();
      dbms_output.put_line('date courante: ' || d1);
      dbms_output.put_line('zone horaie: ' || "DBTIMEZONE"());
      d2 := '23-FEB-2015';
      dbms_output.put_line("TO_CHAR"(d2, 'DAY DD-MM-YYYY'));
      d2 := "NEXT_DAY"(d2, 'SUNDAY');
      dbms_output.put_line("TO_CHAR"(d2, 'DD-MONTH'));
      d1 := '01-AUG-2015';
      dbms_output.put_line("MONTHS_BETWEEN"(d1, d2));
END;
```

Message | DBMS Output

```
01-JAN-02
01/11/2000
date courante: 04-MAR-15
zone horaie: +00:00
MONDAY    23-02-2015
01-MARCH
5
```

4. Manipuler des chaines de caractères

Définition	
ASCII(caractère)	Retourne code ASCII du caractère passé en paramètre.
CHR(entier)	Inverse de **ASCII()**.
COMPOSE(chaine)	Le paramètre chaine doit être comme ceci : *caractère* || *caractère unicode* Retourne une chaine formée de la composition entre le caractère et le caractère unicode. Liste des caractères spéciaux : unistr('\0300') : ` unistr('\0301') : ´ unistr('\0302') : ^ unistr('\0303') : ~ unistr('\0308') : ¨
CONCAT(ch1,ch2)	Le même effet que l'opérateur ||
INITCAP(chaine)	Remplace le 1ier caractère de chaque mot de la chaine par une majuscule, et les autres par des minuscules.
INSTR(ch1,ch2)	Retourne la 1ière occurrence de ch2 dans ch1.
LOWER(chaine)	Transforme toute la chaine en minuscules.
UPPER(chaine)	Transforme toute la chaine en majuscules.
LENGTH(chaine)	Retourne la longueur de la chaine de caractères.
LPAD(ch,n,car)	Rempli la chaine depuis la gauche par le caractère car, jusqu'à atteindre le nombre total de caractères n.
RPAD(ch,n,car)	Rempli la chaine depuis la droite par le caractère car, jusqu'à atteindre le nombre total de caractères n.
LTRIM(chaine)	Enlève les espaces à gauche de la chaine.
LTRIM(chaine, ch)	Enlève toutes les occurrences de la chaine ch à gauche de la chaine.
RTRIM(chaine)	Enlève les espaces à droite de la chaine.
RTRIM(chaine, ch)	Enlève toutes les occurrences de la chaine ch à droite de la

S.ELYAHYAOUI - PL/SQL

	chaine.
TRIM([LEADING | TRAILING | BOTH] ch1 FROM ch2)	Enlève ch1 de la chaine ch2 selon l'option choisie entre **LEADING**, **TRAILING** ou **BOTH**
SUBSTR(chaine, p, l)	Retourne la sous-chaine qui commence à la position p, et qui compte L caractères. Si aucune longueur n'est spécifiée, la fonction retourne tout le reste de la chaine à partir de la position p.
REPLACE(ch1,ch2,ch3)	Remplace la sous-chaine ch2 (2ième paramètre) dans ch1, par la chaine de remplacement ch3 (3ième paramètre). Si aucune chaine de remplacement n'est spécifiée, la sous-chaine ch2 est remplacée par un vide.

Exemple 1

```
BEGIN
      dbms_output.put_line("ASCII"('A'));
      dbms_output.put_line("CHR"('65'));
      dbms_output.put_line("CHR"('6'));
      dbms_output.put_line("CHR"('5'));
      dbms_output.put_line("CHR"('2'));
      dbms_output.put_line("COMPOSE"('o' || unistr('\0308')));
      dbms_output.put_line("INITCAP"('BONJOUR tout le monde'));
      dbms_output.put_line("INSTR"('bonjour tout le monde', 'ou'));
      dbms_output.put_line("UPPER"('BONJOUR tout le monde'));
      dbms_output.put_line("LENGTH"('BONJOUR tout le monde'));
      dbms_output.put_line("LPAD"('test', 6, '*'));
      dbms_output.put_line("RPAD"('test', 6, '*'));
END;
```

Remarque : produire l'effet des lignes **3**, **4** et **5** sur Word (utiliser le raccourci ALT)

Message | DBMS Output

```
65
A
–
|

ö
Bonjour Tout Le Monde
5
BONJOUR TOUT LE MONDE
21
**test
test**
```

Exemple 2

```
BEGIN
      dbms_output.put_line("LTRIM"('     test'));
      dbms_output.put_line("LTRIM"('0000test', '0'));
      dbms_output.put_line("REPLACE"('0000test', '0'));
      dbms_output.put_line("REPLACE"('0000test', '0', '*'));
      dbms_output.put_line("REPLACE"('1212test', '12', '123'));
      dbms_output.put_line("SUBSTR"('Ceci_est_un_test', 2, 8));
      dbms_output.put_line("SUBSTR"('Ceci_est_un_test', 3));
      dbms_output.put_line("TRIM"('    ***bonjour_*    '));
      dbms_output.put_line("TRIM"('*' FROM '    ***bonjour_*'));
      dbms_output.put_line("TRIM"('*' FROM '***bonjour_*'));
      dbms_output.put_line("TRIM"(LEADING '*' FROM '***bonjour_*'));
      dbms_output.put_line("TRIM"(TRAILING '*' FROM '***bonjour_*'));
      dbms_output.put_line("TRIM"(BOTH '*' FROM '***bonjour_*'));
```

```
END;
```

Message	DBMS Output

```
test
test
test
****test
123123test
eci_est_
ci_est_un_test
***bonjour_*
  ***bonjour_
bonjour_
bonjour_*
***bonjour_
bonjour_
```

Exemple 3

Déclarer & initialiser 3 chaines de caractères, et afficher la longueur de la chaine la plus longue et celle de la chaine la plus courte.

```
DECLARE
      d1 VARCHAR2(50);
      d2 VARCHAR2(50);
      d3 VARCHAR2(50);
BEGIN
      d1 := '01-JAN-2001';
      d2 := 'abcd';
      d3 := "CONCAT"(d1, d2);
      dbms_output.put_line("GREATEST"("LENGTH"(d1), "LENGTH"(d2),
"LENGTH"(d3)));
      dbms_output.put_line("LEAST"("LENGTH"(d1), "LENGTH"(d2),
"LENGTH"(d3)));
END;
```

Message	DBMS Output

```
15
4
```

S.ELYAHYAOUI - PLSQL

5. Afficher des données à l'écran

Définition : *PUT_LINE*

Comme on a vu dans les exemples précédents, cette procédure (*aucune valeur de retour*) prédéfinie en PL/SQL dans le paquetage ***DBMS_OUTPUT*** permet d'afficher une ligne avec le rajout d'un caractère de fin de ligne.

(Les notions de fonctions, procédures et packages seront revues en détail dans les prochains chapitres)

Syntaxe :

Opérateur de concaténation des valeurs

```
dbms_output.put_line(val_1 || val_2 || …);
```

Les valeurs peuvent être de type numérique, chaine ou date

Définition : *NEW_LINE*

Cette procédure permet de provoquer un saut de ligne.

Syntaxe :

```
dbms_output.new_line();
```

Exemple

```
BEGIN
      dbms_output.put_line('bonjour');
      dbms_output.new_line();
      dbms_output.put_line('tout le monde');
END;
```

Message	DBMS Output

```
bonjour

tout le monde
```

S.ELYAHYAOUI - PL/SQL